LEXICAL ANALYZER FOR JAVA USING C++

PROJECT REPORT

Submitted for the course: Theory of Computation and Compiler Design (CSE2002)

By

VAIBHAV KHANDELWAL	**15BCE0342**
DIVYANSHI MANGAL	**15BCE0454**
SURAJ KUMAR	**15BCE0647**

Slot F1+TF1

Name of faculty: Prof. D.P. Acharjya

VIT® UNIVERSITY (Estd. u/s 3 of UGC Act 1956) VELLORE ▪ CHENNAI www.vit.ac.in

Index

Objective:

In this project, our objective is to design a lexical analyzer for Java using C++ programming language.

Introduction:

The basic role of a lexical analyzer is to convert a sequence of characters into a sequence of tokens. It removes comments and whitespaces. It reads the characters from the java program, groups them into lexemes and provides us with a sequence of tokens each having a particular meaning in the language.

Overview:

Here we are allowing the user to select a Java file during run time. The program written in C++ by us will analyze the Java code and separate the strings into set of lexemes and accordingly prints the output stating the various keywords, constants, numeric literals, identifiers, operators and etc. from the source code.

- The lexer shall also recognize **identifiers and integer numbers**.

  An *identifier* is a sequence of letters and digits, starting with a letter. The underscore '_' counts as a letter. An integer number is a sequence of digits, possibly starting with a '+' or '_'.

- For each identifier, the lexer shall return the token IDENTIFIER, and for each integer number, it shall return the token NUMERIC LITERAL.

- The lexer shall also recognize the operators used in Java.

- **The lexer shall recognize all of the following 50 Java keywords:**

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Project Resource Requirements:

- Dev C++ / CodeBlocks IDE
- BlueJ IDE

Work Break-down:

Team Member Registration Number	Name	Work Assigned and Done
15BCE0342	Vaibhav Khandelwal	C++ Code, Algorithm, Documentation, Code Debugging and Testing.
15BCE0454	Divyanshi Mangal	C++ Code, Documentation,Code Debugging and Testing.
15BCE0647	Suraj Kumar	Sample JAVA files, Pseudo Code

Algorithm:

1. Keywords, Seperators, Operators etc. of JAVA are defined and stored in array.

2. User is prompted to enter the JAVA file location.

3. Once user enters the file location, the program validates the presence of file. If file is not present, an error is displayed saying "File not found!" and the program terminates. If file is present, then it proceeds to step 4.

4. After validating the file, it analyzes the entire JAVA file and matches the tokens with the operators, identifiers, keywords, header files etc. which are already stored in the array. For identification of numbers, it takes the help of “isdigit (int)” function defined under “#include<ctype.h>” header file.

5. After successful analysis, it displays the identified tokens.

6. This program also displays the overall statistics of JAVA file, i.e. it counts the number of various tokens. The number of tokens were also counted manually and were found same as those displayed by the program which proves the correctness of code.

7. On successful completion, the program terminates.

Pseudo Code:

```
START
CHAR keywords[50][15]={"abstract","assert","boolean","break",
                                        "byte","catch","case","char","const",
                                        "continue","class","default","do",
                                        "double","else","enum","extends",
                                        "final","finally","float","for",
                                        "goto","if","import","implements",
                                        "int","interface","insatnceof","long",
                                        "native","new","package","private",
                                        "public","protected","return","short",
                                        "static","switch","strictfp","super",
                                  "synchronized","this","throw","transient",
                                  "try","volatile","void","while"};



SET INT i=0,count_identifier=0,count_keyword=0,count_number=0,count_operator=0;

VOID  check_key(char *p){
       INT k
       SET flag=0;
       FOR k = 0 TO 48{
              IF COMPARE STRINGS (keywords[k], p)==0)
              THEN
                     PRINT  "P is a keyword in given JAVA file.. "
                     count_keyword++;
                     SET flag=1
                     BREAK
              END IF
END FOR

       IF flag = 0
```

```
            THEN
                  IF p[0]  IS NUMBER
                  THEN
                        PRINT  "P[0]is a number in given JAVA file"
                        count_number++;
                  ELSE
                        IF p[0] IS NOT NULL
                        THEN
                              PRINT  "p s is an identifier in given JAVA file.."
                              count_identifier++;
                        END IF
                  END IF
      END IF
DECREMENT i BY 1



INT main() {
      INT j;
      CHAR filename[50];
      CHAR chr,check_str[25],separators[20]=" \n\t,;(){}[]#\"<>",oprators[20]="!%^&*-
+=~|.<>/?";
      FILE *f1;
      PRINT "Welcome.. This is Lexical Analysis..
            Kindly Enter the JAVA file location: (drive:\\folder\\filename) "
      INPUT file patH
      SET f1 = OPENFILE(filename,"r");
      IF file IS not found;
      THEN
            PRINT "OOPS! File not found...! "
            EXIT from program
      ENDIF
      WHILE ((chr = get_a_character(f1))! = EndOfFile)
      DO
            FOR J = 0 TO 14
                  IF chr IS An operator
                  THEN
                        PRINT “chr is an operator in given JAVA file”
                        count_operator++;
                        SET ckeck_str[i]='NULL
                        check_key(ckeck_str);
                  END IF
      END FOR
      FOR j = 0 TO j = 14
            IF i EQUALS -1
            THEN
                  BREAK
      IF chr IS in separators
      THEN
            If chr=='#')
            THEN
                  While chr IS NOT EQUAL  >
                        PRINT “chr”
                        chr=fgetc(f1);
                  END WHILE
                  PRINT “chr is a header file in JAVA”
                  i=-1;
                  BREAK;
      END IF
      IF chr IS “
      THEN
            DO
                  chr=get_a_character(f1);
                              PRINT “chr”
                        WHILE  (chr IS not  “ )
                        END DO WHILE
                              PRINT " chr is an argument in given JAVA file"
                              i=-1;
```

```
                                        break;
                                END IF
                        ckeck_str[i] IS NULL
                        check_key(ckeck_str);
                END IF
        END FOR
        IF i NOT EQUAL -1
        THEN
                ckeck_str[i]=chr;
                i++;
        END IF
        ELSE
                SET i=0
        PRINT "______________________________________________________\n"
        PRINT "Overall Statistics of given JAVA program"
        PRINT "*********No. of Keywords in JAVA file are count_keyword"
        PRINT "No. of Identifiers in JAVA file are:"count_identifier"
        PRINT "No. of Operators in JAVA file are: count_operator"
        PRINT"No. of Numeric literals in JAVA file arecount_number"
        PRINT "*********"
END
```

Source Code in C++ :

```
//lex_java.cpp

//lexical Analyzer for java

//author@ 15BCE0342,15BCE0454


#include<stdio.h>

#include<string.h>

#include<ctype.h>              //for testing and mapping characters...

#include<iostream>

#include<stdlib.h>

#include<windows.h>


using namespace std;


char keywords[50][15]={"abstract","assert","boolean","break",

                    "byte","catch","case","char","const",

                    "continue","class","default","do",

                    "double","else","enum","extends",

                    "final","finally","float","for",

                    "goto","if","import","implements",

                    "int","interface","insatnceof","long",

                    "native","new","package","private",

                    "public","protected","return","short",

                    "static","switch","strictfp","super",

                    "synchronized","this","throw","throws","transient",

                    "try","volatile","void","while"};
```

```
int
i=0,count_identifier=0,count_keyword=0,count_number=0,count_operator=0,count_separa
tors=0;

void check_key(char *p)

{

int k,flag=0;

for(k=0;k<=48;k++)

{

if(strcmp(keywords[k],p)==0)

{

printf("%s :- KEYWORD \n",p);   //for checking if given character is keyword...

Sleep(50);

count_keyword++;

flag=1;

break;

}

}

if(flag==0)

{

if(isdigit(p[0]))

{

printf("%s :- NUMERIC LITERAL \n",p);  //for checking if given character is Numeric
Literal...

Sleep(50);

count_number++;

}

else

{

if(p[0]!='\0')

{
```

```
printf("%s :- IDENTIFIER\n",p);   //for checking if given character is
identifier...

Sleep(50);

count_identifier++;

}

}

}

i=-1;

}



int main()

{   int j;

    char filename[50];

    char chr,ckeck_str[25],separators[20]="
\n\t,;:(){}[]#\"<>",oprators[20]="!%^&*-+=~|.<>/?";

    FILE *f1;



cout<<"Welcome.. This is Lexical Analysis..\n\nKindly Enter the JAVA file location:
(drive:\\folder\\filename) \n";

cin>>ws;

cin.getline(filename,50);



f1 = fopen(filename,"r");



    if(f1==NULL)

    {

     cout<<"OOPS! File not found...! ";

     exit(0);

    }
```

```
cout<<"\n\nAnalysing...";

cout<<filename<<"\nPlease wait...\n\n";

Sleep(1000);

cout<<"Identifying tokens...\n";

Sleep(1000);

cout<<"Identifying identifiers...\n";

Sleep(1000);

cout<<"Identifying operators...\n";

Sleep(1000);

cout<<"Identifying numeric literals...\n";

Sleep(1000);

cout<<"File scan Completed Successfully...\a\n";

Sleep(600);

cout<<"Displaying all tokens...\n\n\a";

Sleep(1000);

cout<<"******************************************\n";


while((chr=fgetc(f1))!=EOF)

{

for(j=0;j<=14;j++)

{

if(chr==oprators[j])     //for checking if given character is operator...

{

cout<<chr<<" :- OPERATOR \n";

Sleep(50);

count_operator++;

ckeck_str[i]='\0';

check_key(ckeck_str);

}
```

```
}

for(j=0;j<=14;j++)

{

if(i==-1)

break;

if(chr==separators[j])  //for checking if given character is separator...

{


if(chr==';'||chr==','||chr=='<'||chr=='>'||chr=='{'||chr=='}'||chr=='('||chr==')'||
chr==':')

    {cout<<chr<<" :- SEPARATOR\n";

    count_separators++;

    Sleep(50);}


if(chr=='#')

{

while(chr!='>')

{

printf("%c",chr);

chr=fgetc(f1);

}

i=-1;

break;

}

if(chr=='"')

{

do

{

chr=fgetc(f1);

printf("%c",chr);
}while(chr!='"');

cout<<"\b"<<" :- ARGUMENT \n";   //for checking if given character is argument to a
function...

Sleep(50);

i=-1;

break;

}

ckeck_str[i]='\0';

check_key(ckeck_str);

}

}

if(i!=-1)

{

ckeck_str[i]=chr;

i++;

}

else

i=0;

    }

cout<<"______________________________________________________\n";

cout<<"\nOverall Statistics of given JAVA program.\a\n";

cout<<"\n\n*********\nNo. of Keywords in JAVA file are: "<<count_keyword;

cout<<"\nNo. of Identifiers in JAVA file are: "<<count_identifier;

cout<<"\nNo. of Operators in JAVA file are: "<<count_operator;

cout<<"\nNo. of Numeric literals in JAVA file are: "<<count_number;

cout<<"\nNo. of Separators in JAVA file are: "<<count_separators;

cout<<"\n*********\n";

return 0;

}
```

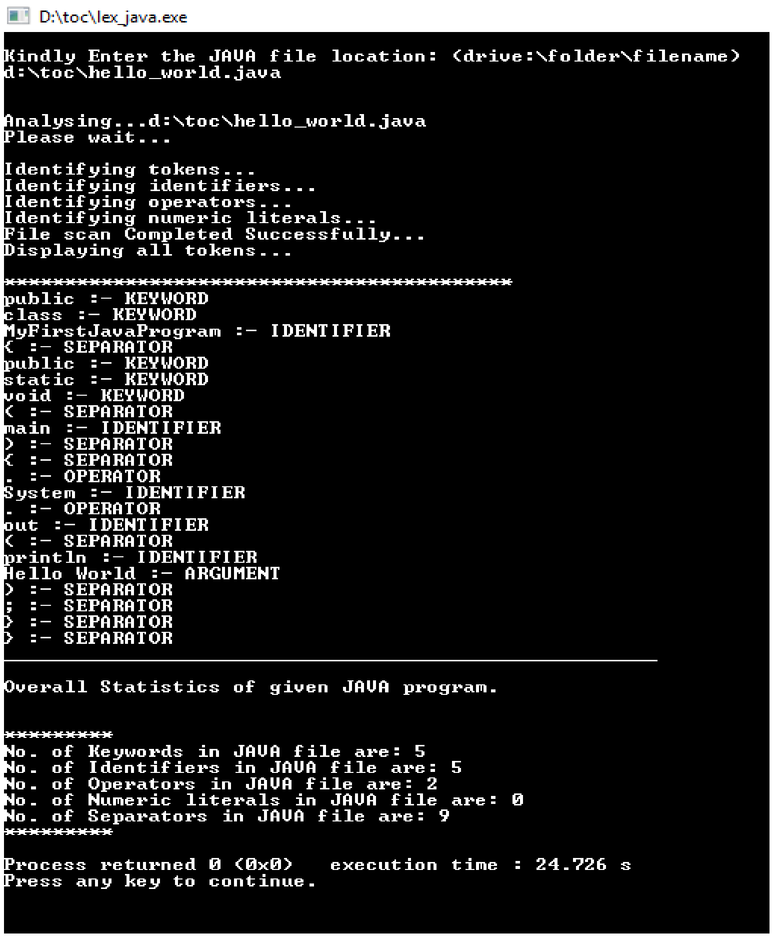
Sample JAVA Programs and Test Cases:

The program was checked for about 20 JAVA programs and it worked fine and passed all of the test cases. Few sample test cases are presented below:

1. hello_world.java

```
public class MyFirstJavaProgram {

   public static void main() {

      System.out.println("Hello World");

   }

}
```

Output:

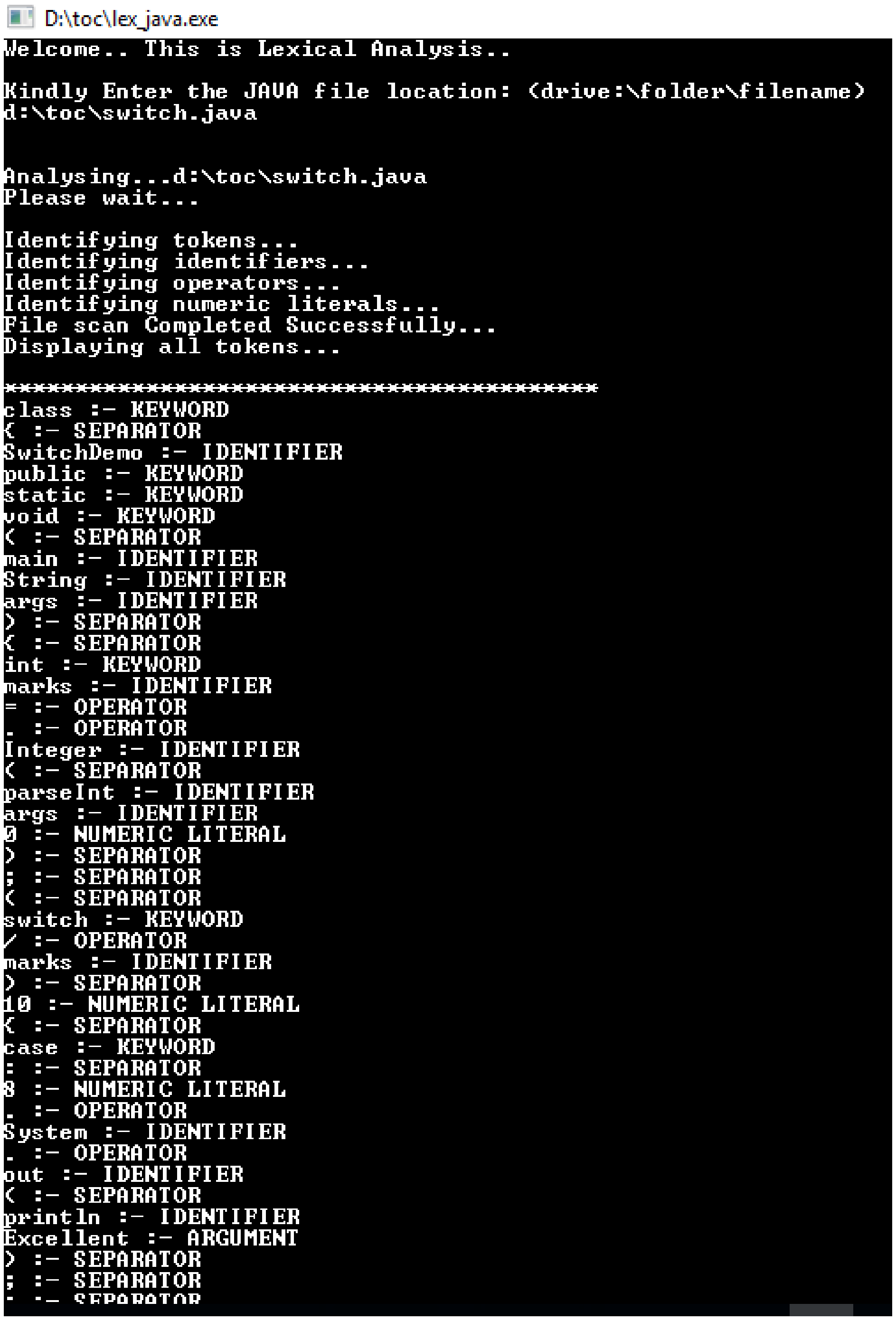


```
D:\toc\lex_java.exe
Kindly Enter the JAVA file location: (drive:\folder\filename)
d:\toc\hello_world.java


Analysing...d:\toc\hello_world.java
Please wait...

Identifying tokens...
Identifying identifiers...
Identifying operators...
Identifying numeric literals...
File scan Completed Successfully...
Displaying all tokens...

******************************************
public :- KEYWORD
class :- KEYWORD
MyFirstJavaProgram :- IDENTIFIER
{ :- SEPARATOR
public :- KEYWORD
static :- KEYWORD
void :- KEYWORD
( :- SEPARATOR
main :- IDENTIFIER
) :- SEPARATOR
{ :- SEPARATOR
. :- OPERATOR
System :- IDENTIFIER
. :- OPERATOR
out :- IDENTIFIER
( :- SEPARATOR
println :- IDENTIFIER
Hello World :- ARGUMENT
) :- SEPARATOR
; :- SEPARATOR
} :- SEPARATOR
} :- SEPARATOR
______________________________________________________

Overall Statistics of given JAVA program.


*********
No. of Keywords in JAVA file are: 5
No. of Identifiers in JAVA file are: 5
No. of Operators in JAVA file are: 2
No. of Numeric literals in JAVA file are: 0
No. of Separators in JAVA file are: 9
*********

Process returned 0 (0x0)   execution time : 24.726 s
Press any key to continue.
```

2. switch.java

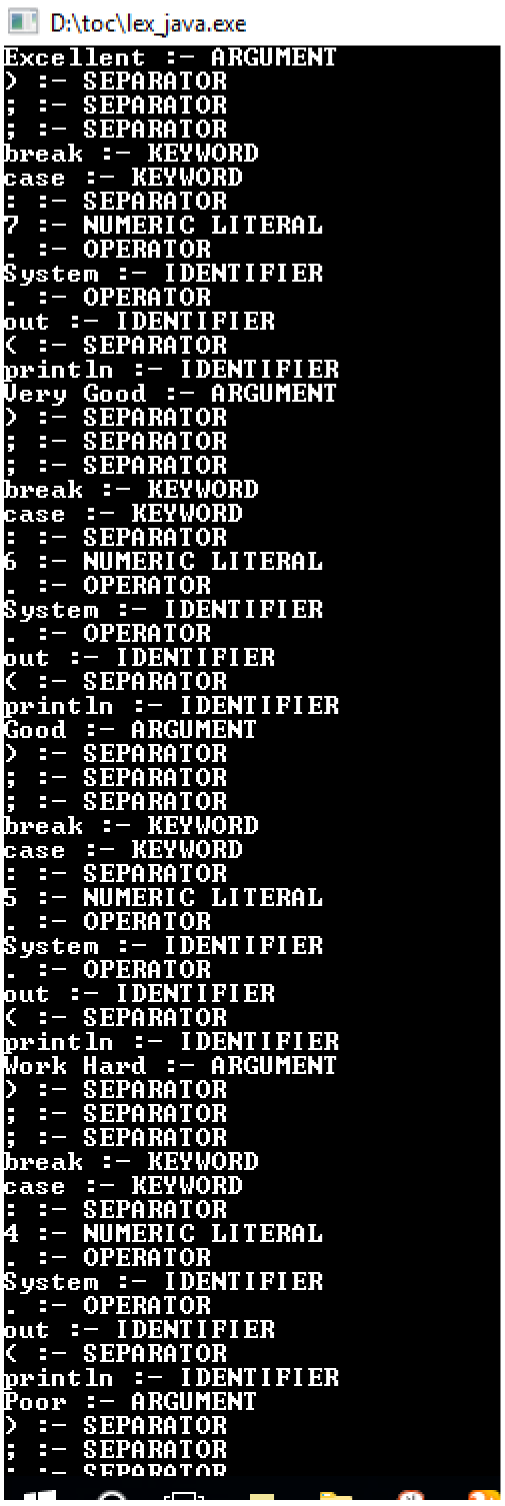
```
class SwitchDemo{

      public static void main(String args[]){

          int marks = Integer.parseInt(args[0]);

       switch(marks/10){

             case 8:
                      System.out.println("Excellent");
                      break;
             case 7:
                      System.out.println("Very Good");
                      break;
             case 6:
                      System.out.println("Good");
                      break;
             case 5:
                      System.out.println("Work Hard");
                      break;
             case 4:
                      System.out.println("Poor");
                      break;

             case 0:
                      System.out.println("Very Poor");
                      break;
             default :

                      System.out.println("Invalid value Entered");

      }

 }

}
```

Output:

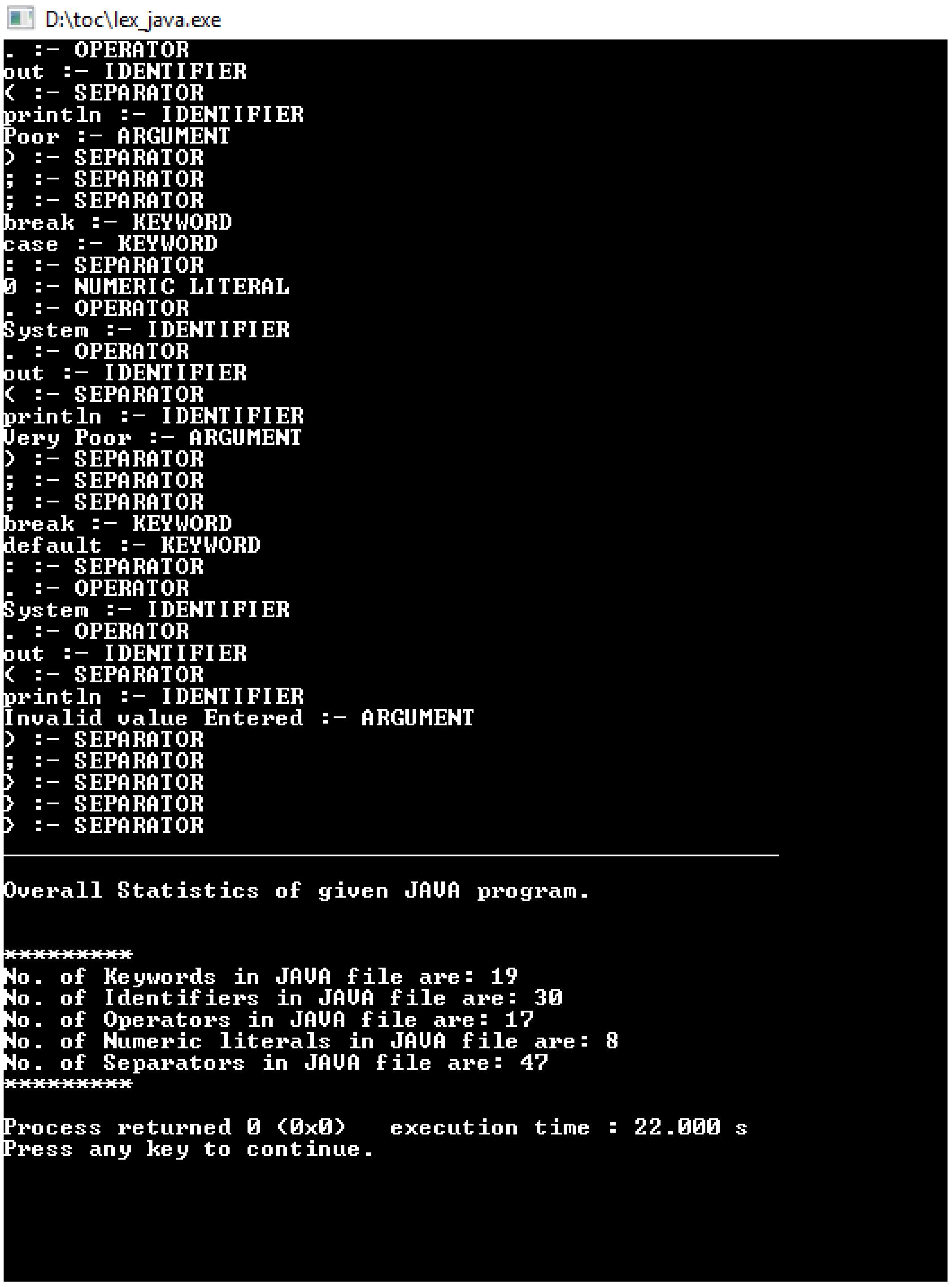


D:\toc\lex_java.exe

```
Welcome.. This is Lexical Analysis..

Kindly Enter the JAVA file location: (drive:\folder\filename)
d:\toc\switch.java


Analysing...d:\toc\switch.java
Please wait...

Identifying tokens...
Identifying identifiers...
Identifying operators...
Identifying numeric literals...
File scan Completed Successfully...
Displaying all tokens...

******************************************
class :- KEYWORD
{ :- SEPARATOR
SwitchDemo :- IDENTIFIER
public :- KEYWORD
static :- KEYWORD
void :- KEYWORD
( :- SEPARATOR
main :- IDENTIFIER
String :- IDENTIFIER
args :- IDENTIFIER
) :- SEPARATOR
{ :- SEPARATOR
int :- KEYWORD
marks :- IDENTIFIER
= :- OPERATOR
. :- OPERATOR
Integer :- IDENTIFIER
( :- SEPARATOR
parseInt :- IDENTIFIER
args :- IDENTIFIER
0 :- NUMERIC LITERAL
) :- SEPARATOR
; :- SEPARATOR
( :- SEPARATOR
switch :- KEYWORD
/ :- OPERATOR
marks :- IDENTIFIER
) :- SEPARATOR
10 :- NUMERIC LITERAL
{ :- SEPARATOR
case :- KEYWORD
: :- SEPARATOR
8 :- NUMERIC LITERAL
. :- OPERATOR
System :- IDENTIFIER
. :- OPERATOR
out :- IDENTIFIER
( :- SEPARATOR
println :- IDENTIFIER
Excellent :- ARGUMENT
) :- SEPARATOR
; :- SEPARATOR
: :- SEPARATOR
```


D:\toc\lex_java.exe

```
Excellent :- ARGUMENT
) :- SEPARATOR
; :- SEPARATOR
; :- SEPARATOR
break :- KEYWORD
case :- KEYWORD
: :- SEPARATOR
7 :- NUMERIC LITERAL
. :- OPERATOR
System :- IDENTIFIER
. :- OPERATOR
out :- IDENTIFIER
( :- SEPARATOR
println :- IDENTIFIER
Very Good :- ARGUMENT
) :- SEPARATOR
; :- SEPARATOR
; :- SEPARATOR
break :- KEYWORD
case :- KEYWORD
: :- SEPARATOR
6 :- NUMERIC LITERAL
. :- OPERATOR
System :- IDENTIFIER
. :- OPERATOR
out :- IDENTIFIER
( :- SEPARATOR
println :- IDENTIFIER
Good :- ARGUMENT
) :- SEPARATOR
; :- SEPARATOR
; :- SEPARATOR
break :- KEYWORD
case :- KEYWORD
: :- SEPARATOR
5 :- NUMERIC LITERAL
. :- OPERATOR
System :- IDENTIFIER
. :- OPERATOR
out :- IDENTIFIER
( :- SEPARATOR
println :- IDENTIFIER
Work Hard :- ARGUMENT
) :- SEPARATOR
; :- SEPARATOR
; :- SEPARATOR
break :- KEYWORD
case :- KEYWORD
: :- SEPARATOR
4 :- NUMERIC LITERAL
. :- OPERATOR
System :- IDENTIFIER
. :- OPERATOR
out :- IDENTIFIER
( :- SEPARATOR
println :- IDENTIFIER
Poor :- ARGUMENT
) :- SEPARATOR
; :- SEPARATOR
; :- SEPARATOR
```



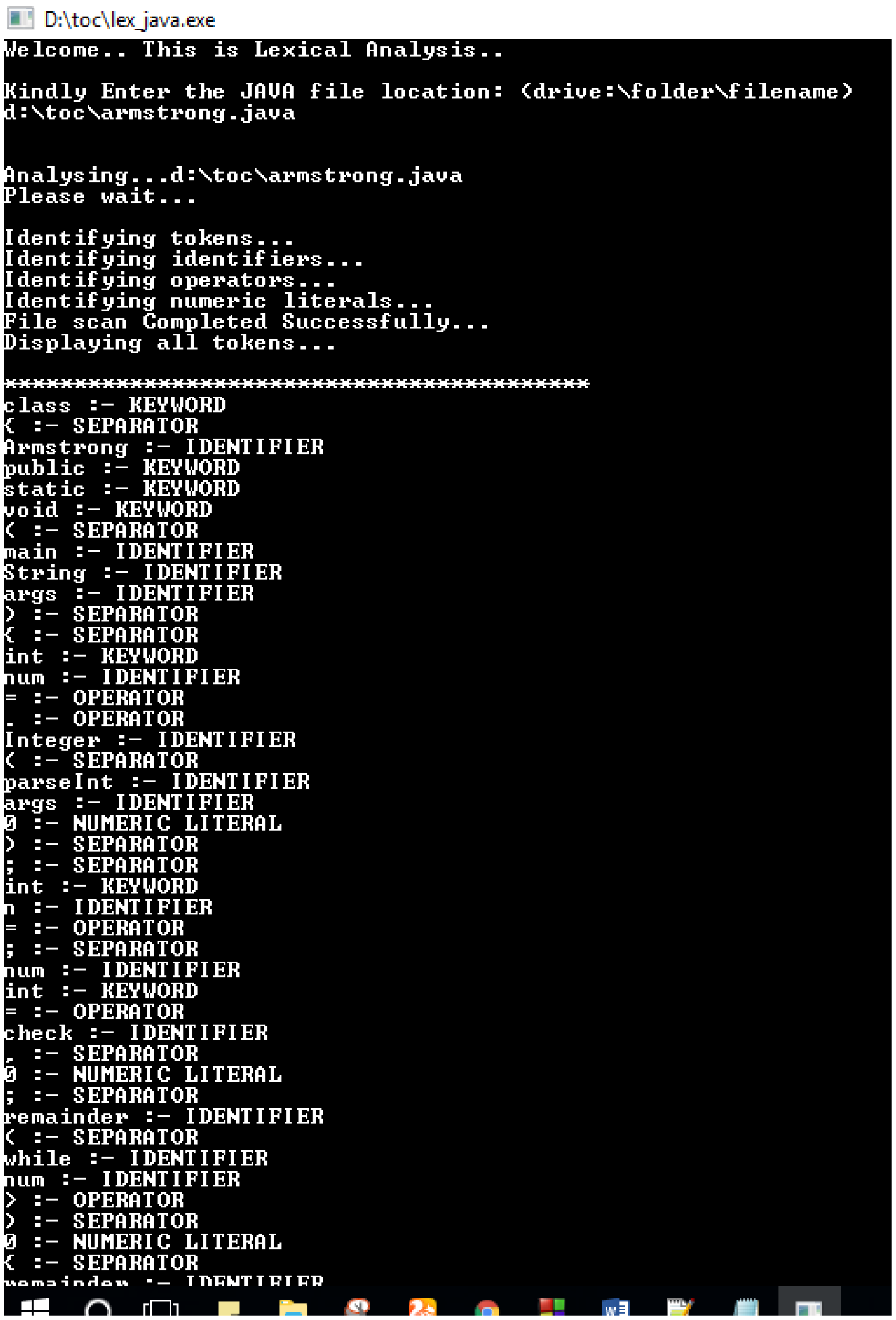
D:\toc\lex_java.exe

```
. :- OPERATOR
out :- IDENTIFIER
( :- SEPARATOR
println :- IDENTIFIER
Poor :- ARGUMENT
) :- SEPARATOR
; :- SEPARATOR
; :- SEPARATOR
break :- KEYWORD
case :- KEYWORD
: :- SEPARATOR
0 :- NUMERIC LITERAL
. :- OPERATOR
System :- IDENTIFIER
. :- OPERATOR
out :- IDENTIFIER
( :- SEPARATOR
println :- IDENTIFIER
Very Poor :- ARGUMENT
) :- SEPARATOR
; :- SEPARATOR
; :- SEPARATOR
break :- KEYWORD
default :- KEYWORD
: :- SEPARATOR
. :- OPERATOR
System :- IDENTIFIER
. :- OPERATOR
out :- IDENTIFIER
( :- SEPARATOR
println :- IDENTIFIER
Invalid value Entered :- ARGUMENT
) :- SEPARATOR
; :- SEPARATOR
} :- SEPARATOR
} :- SEPARATOR
} :- SEPARATOR
______________________________________________________

Overall Statistics of given JAVA program.


**********
No. of Keywords in JAVA file are: 19
No. of Identifiers in JAVA file are: 30
No. of Operators in JAVA file are: 17
No. of Numeric literals in JAVA file are: 8
No. of Separators in JAVA file are: 47
**********

Process returned 0 (0x0)   execution time : 22.000 s
Press any key to continue.
```

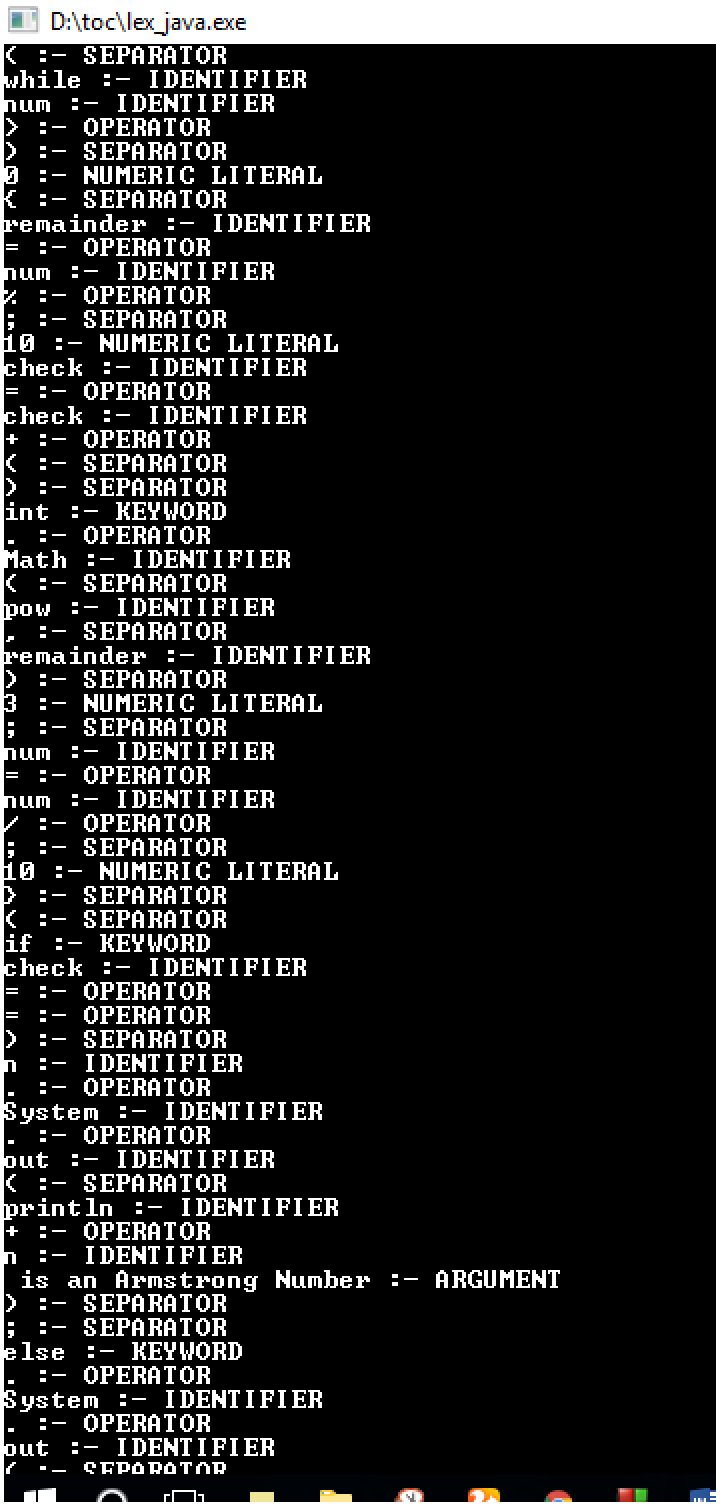
3. armstrong.java

```
class Armstrong{

      public static void main(String args[]){

      int num = Integer.parseInt(args[0]);

      int n = num;

      int check=0,remainder;

      while(num > 0){

           remainder = num % 10;

           check = check + (int)Math.pow(remainder,3);

           num = num / 10;

      }

      if(check == n)

            System.out.println(n+" is an Armstrong Number");

      else

            System.out.println(n+" is not a Armstrong Number");

   }

}
```

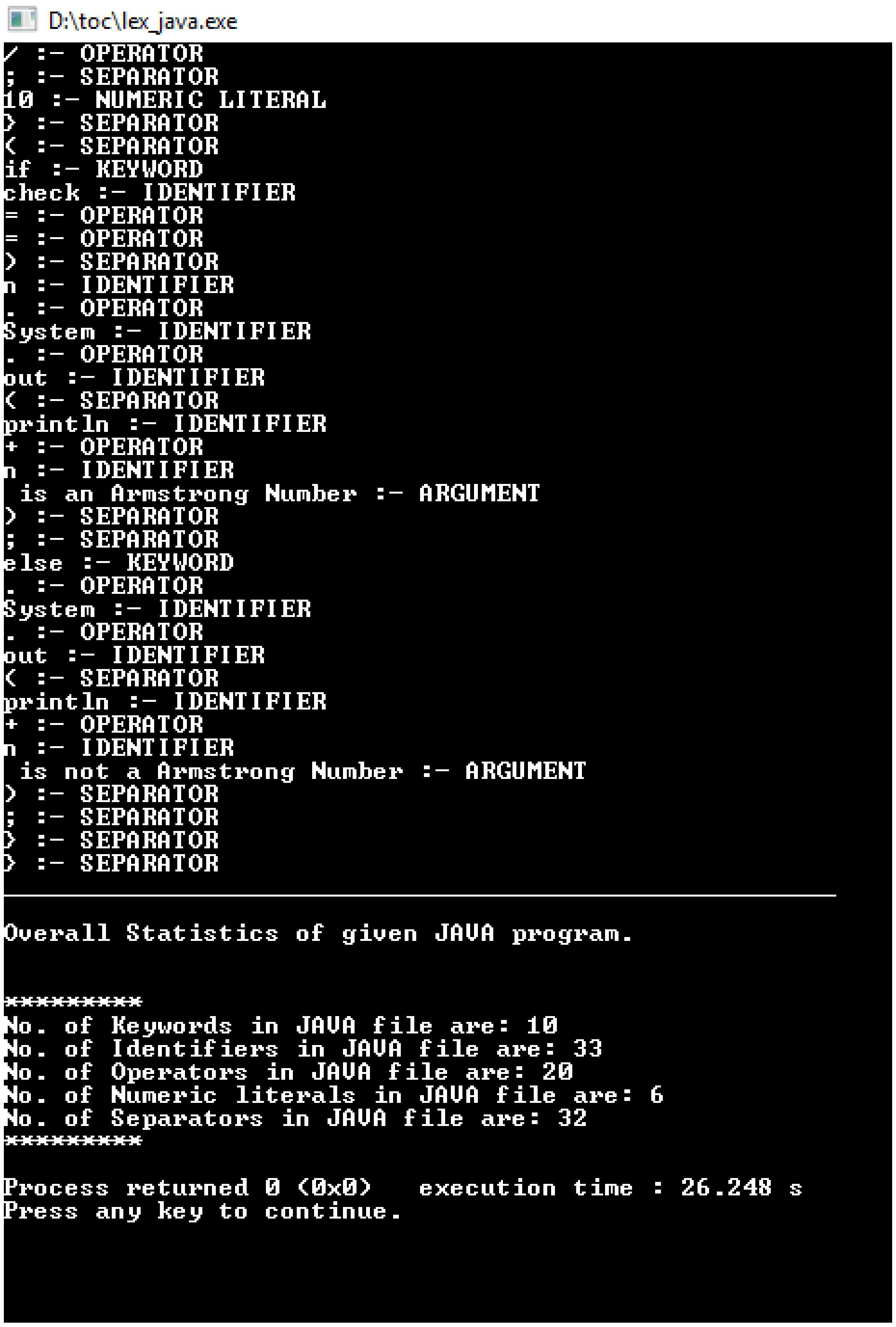
Output:



```
D:\toc\lex_java.exe
Welcome.. This is Lexical Analysis..

Kindly Enter the JAVA file location: (drive:\folder\filename)
d:\toc\armstrong.java


Analysing...d:\toc\armstrong.java
Please wait...

Identifying tokens...
Identifying identifiers...
Identifying operators...
Identifying numeric literals...
File scan Completed Successfully...
Displaying all tokens...

******************************************
class :- KEYWORD
{ :- SEPARATOR
Armstrong :- IDENTIFIER
public :- KEYWORD
static :- KEYWORD
void :- KEYWORD
( :- SEPARATOR
main :- IDENTIFIER
String :- IDENTIFIER
args :- IDENTIFIER
) :- SEPARATOR
{ :- SEPARATOR
int :- KEYWORD
num :- IDENTIFIER
= :- OPERATOR
. :- OPERATOR
Integer :- IDENTIFIER
( :- SEPARATOR
parseInt :- IDENTIFIER
args :- IDENTIFIER
0 :- NUMERIC LITERAL
) :- SEPARATOR
; :- SEPARATOR
int :- KEYWORD
n :- IDENTIFIER
= :- OPERATOR
; :- SEPARATOR
num :- IDENTIFIER
int :- KEYWORD
= :- OPERATOR
check :- IDENTIFIER
, :- SEPARATOR
0 :- NUMERIC LITERAL
; :- SEPARATOR
remainder :- IDENTIFIER
( :- SEPARATOR
while :- IDENTIFIER
num :- IDENTIFIER
> :- OPERATOR
) :- SEPARATOR
0 :- NUMERIC LITERAL
{ :- SEPARATOR
remainder :- IDENTIFIER
```


D:\toc\lex_java.exe

```
( :- SEPARATOR
while :- IDENTIFIER
num :- IDENTIFIER
> :- OPERATOR
) :- SEPARATOR
0 :- NUMERIC LITERAL
( :- SEPARATOR
remainder :- IDENTIFIER
= :- OPERATOR
num :- IDENTIFIER
% :- OPERATOR
; :- SEPARATOR
10 :- NUMERIC LITERAL
check :- IDENTIFIER
= :- OPERATOR
check :- IDENTIFIER
+ :- OPERATOR
( :- SEPARATOR
) :- SEPARATOR
int :- KEYWORD
. :- OPERATOR
Math :- IDENTIFIER
( :- SEPARATOR
pow :- IDENTIFIER
, :- SEPARATOR
remainder :- IDENTIFIER
) :- SEPARATOR
3 :- NUMERIC LITERAL
; :- SEPARATOR
num :- IDENTIFIER
= :- OPERATOR
num :- IDENTIFIER
/ :- OPERATOR
; :- SEPARATOR
10 :- NUMERIC LITERAL
) :- SEPARATOR
( :- SEPARATOR
if :- KEYWORD
check :- IDENTIFIER
= :- OPERATOR
= :- OPERATOR
) :- SEPARATOR
n :- IDENTIFIER
. :- OPERATOR
System :- IDENTIFIER
. :- OPERATOR
out :- IDENTIFIER
( :- SEPARATOR
println :- IDENTIFIER
+ :- OPERATOR
n :- IDENTIFIER
 is an Armstrong Number :- ARGUMENT
) :- SEPARATOR
; :- SEPARATOR
else :- KEYWORD
. :- OPERATOR
System :- IDENTIFIER
. :- OPERATOR
out :- IDENTIFIER
( :- SEPARATOR
```



D:\toc\lex_java.exe

```
/ :- OPERATOR
; :- SEPARATOR
10 :- NUMERIC LITERAL
) :- SEPARATOR
( :- SEPARATOR
if :- KEYWORD
check :- IDENTIFIER
= :- OPERATOR
= :- OPERATOR
) :- SEPARATOR
n :- IDENTIFIER
. :- OPERATOR
System :- IDENTIFIER
. :- OPERATOR
out :- IDENTIFIER
( :- SEPARATOR
println :- IDENTIFIER
+ :- OPERATOR
n :- IDENTIFIER
 is an Armstrong Number :- ARGUMENT
) :- SEPARATOR
; :- SEPARATOR
else :- KEYWORD
. :- OPERATOR
System :- IDENTIFIER
. :- OPERATOR
out :- IDENTIFIER
( :- SEPARATOR
println :- IDENTIFIER
+ :- OPERATOR
n :- IDENTIFIER
 is not a Armstrong Number :- ARGUMENT
) :- SEPARATOR
; :- SEPARATOR
} :- SEPARATOR
} :- SEPARATOR

Overall Statistics of given JAVA program.


**********
No. of Keywords in JAVA file are: 10
No. of Identifiers in JAVA file are: 33
No. of Operators in JAVA file are: 20
No. of Numeric literals in JAVA file are: 6
No. of Separators in JAVA file are: 32
**********

Process returned 0 (0x0)   execution time : 26.248 s
Press any key to continue.
```

References:

- J.P. Bennett, Introduction to Compiling Techniques
- J. Hopcroft, J. Ullman, Introduction to Automata Theory, Languages, and Computation
- https://www.tutorialspoint.com/java/
- http://beginnersbook.com/java-tutorial-for-beginners-with-examples/
- http://stackoverflow.com/questions/17848207/making-a-lexical-analyzer